an argument to a function. In this case, the scene itself has an argument that specifies what company the user is looking at and the scene is accordingly customized. Thus, when the user looks through any of these wormholes, the scene looks different because it takes on the identity of the specific wormhole being viewed by the user.

As discussed above, the system provides dynamic views of data without programming expertise. Users are thus moved closer to the data so that application development time is reduced. User interfaces may be created quickly and easily for information rich databases and for applications such as data warehousing and decision support. Further, limitations inherent in conventional forms-based or report-based applications are avoided.

Moreover, the techniques described here may be implemented in hardware or software, or a combination of the two. Preferably, the techniques are implemented in computer programs executing on programmable computers that each includes a processor, a storage medium readable by the processor (including volatile and nonvolatile memory and/or storage elements), and suitable input and output devices. Program code is applied to data entered using an input device to perform the functions described and to generate output information. The output information is applied to one or more output devices.

Each program is preferably implemented in a high level procedural or object-oriented programming language to communicate with a computer system. However, the programs can be implemented in assembly or machine language, if desired. In any case, the language may be a compiled or interpreted language.

Each such computer program is preferably stored on a storage medium or device (e.g., CD-ROM, hard disk or magnetic diskette) that is readable by a general or special purpose programmable computer for configuring and operating the computer when the storage medium or device is read by the computer to perform the procedures described. The system also may be implemented as a computer-readable storage medium, configured with a computer program, where the storage medium so configured causes a computer to operate in a specific and predefined manner.

Other embodiments are within the scope of the following claims.

What is claimed is:

1. A method for executing an application expressed as a scene graph, the scene graph being a hierarchical representation of one or more objects, each object capable of generating code associated with the object, comprising:

traversing the hierarchy of the scene graph;

retrieving byte code associated with each object in the scene graph;

characterizing each object as one of several types, including a shape type object, and for each shape type object generating a create shape statement;

storing the byte code as part of the scene;

generating the byte code execution image for the scene;

providing the byte code image to an execution engine and executing each statement in the byte code;

determining whether each statement is a create shape statement, and if so creating the shape; and

storing the shape in a context hash table using a tokenized shape name.

2. The method of claim **1**, further comprising determining whether the statement is a begin property statement, and if so:

searching for the shape in a context hash table; and

executing a begin method associated with the shape.

3. The method of claim **1**, further comprising determining whether the statement is a set property statement, and if so:

searching for the shape in a context hash table;

evaluating a property expression associated with the shape; and

assigning a value to the property.

4. The method of claim **1**, further comprising determining whether the statement is an end property statement, and if so:

searching for the shape in a context hash table;

executing an end properties method associated with the shape to commit the properties;

initializing the shape; and

adding the shape to a display list.

5. Computer software for executing an application expressed as a scene graph, the scene graph being a hierarchical representation of one or more objects, each object capable of generating code associated with the object, the computer software residing on a computer-readable medium and comprising instructions for causing a computer to perform the following operations:

traverse the hierarchy of the scene graph;

retrieve byte code associated with each object in the scene graph;

characterize each object as one of several types, including a shape type object, and for each shape type object generate a create shape statement;

store the byte code as part of the scene;

generate the byte code execution image for the scene;

provide the byte code image to an execution engine and execute each statement in the byte code;

determine whether each statement is a create shape statement, and if so create the shape; and

store the shape in a context hash table using a tokenized shape name.

6. The computer software of claim **5**, further comprising instructions to determine whether the statement is a begin property statement, and if so:

search for the shape in a context hash table; and

execute a begin method associated with the shape.

7. The computer software of claim **5**, further comprising instructions to determine whether the statement is a set property statement, and if so:

search for the shape in a context hash table;

evaluate a property expression associated with the shape; and

assign a value to the property.

8. The computer software of claim **5**, further comprising instructions to determine whether the statement is an end property statement, and if so:

search for the shape in a context hash table;

execute an end properties method associated with the shape to commit the properties;

initialize the shape; and

add the shape to a display list.

* * * * *